Combining DEVS and model-checking: concepts and tools for integrating simulation and analysis

Bernard P. Zeigler*

RTSync Corporation,
Arizona Center for Integrative Modelling and Simulation,
AZ, USA
Email: zeigler@rtsync.com
*Corresponding author

James J. Nutaro

OakRidge National Laboratory,
One Bethel Valley Road, Oak Ridge,
TN 37831, USA
Email: nutarojj@ornl.gov

Chungman Seo

RTSync Corporation,
Arizona Center for Integrative Modelling and Simulation,
AZ, USA
Email: cseo@rtsync.com

Abstract: Our objectives here are to discuss the development of a formal framework that exploits the advantages of the discrete event system specification (DEVS) formalism and builds upon recent extensive work on verification combining DEVS and model checking for hybrid systems. DEVS offers the ability, via mathematical transformations called system morphisms, to map a system expressed in a formalism suitable for analysis (e.g., timed automata or hybrid automata) into the DEVS formalism for the purpose of simulation. We discuss a probabilistic extension of the FD-DEVS formalism that enables a set of model classes and tools derived from Markov-type models. The MS4 modelling environment provides a suite of tools that support this extension, called FP-DEVS. In this paper, we describe these tools and the concepts underlying them. We also provide examples of application of these concepts and discuss the open opportunities for research in this direction.

Keywords: model checking; verification and validation; cyber-physical systems; autonomous systems; cooperative systems; systems of systems; modelling and simulation; general dynamic systems theory; discrete event system specification; DEVS; combining simulation and formal verification; Markov models: Lumpable Markov chains.

Reference to this paper should be made as follows: Zeigler, B.P., Nutaro, J.J. and Seo, C. (2017) 'Combining DEVS and model-checking: concepts and tools for integrating simulation and analysis', *Int. J. Simulation and Process Modelling*, Vol. 12, No. 1, pp.2–15.

Biographical notes: Bernard P. Zeigler is a Chief Scientist of RTSync Corporation and Emeritus Professor from Arizona Center for Integrative Modelling and Simulation. He is internationally known for his seminal contributions in M&S theory, and has published several books including *Theory of Modelling and Simulation*. IEEE named him a fellow of the IEEE for his invention of the discrete event system specification (DEVS).

James J. Nutaro is senior research staff at Oak Ridge National Laboratory. He has extensive experience in M&S and systems modelling in both defense and commercial domains. He has applied M&S techniques for design, analysis, and testing in diverse enterprises including missile systems, space systems, communications, uranium processing, electrical power, disease processes, and high performance computing technology. He is also the author of *Building Software for Simulation: Theory and Algorithms, with Applications in C++*.

Chungman Seo is a senior research engineer at MS4 Systems Company and a member of Arizona Center for Integrative Modelling and Simulation (ACIMS). He received his PhD in Electrical and Computer Engineering from The University of Arizona in 2009. His research includes MS4 Me Product, web-based DEVS simulation system, DEVS/SOA-based distributed DEVS simulation, and DEVS simulator interoperability.

This paper is a revised and expanded version of a paper entitled 'Combining DEVS and model-checking: using systems morphisms for integrating simulation and analysis in model engineering' presented at EMSS: European Modelling & Simulation Symposium, 26th edition, Part of I3M2014, Bordeaux, France, 10–12 September 2014.

1 Introduction

Model checking, a well-known formal verification method, systematically explores the state space of a system model to check that states satisfy specified behavioural properties (Baier and Joost-Pieter, 2008). Model checking methods encounter state space explosion in analysing *autonomous* systems that require complex logical processes to perform complex decision making tasks. Moreover, because they are limited in their expressive capability to restricted logics, such methods must typically make stringent assumptions about physical components and environments. These assumptions and idealisations greatly reduce the methods' applicability to *cyber-physical systems* where the interplay of physical and computational elements is paramount. Finally, *cooperative* multi-agent systems raise the state space explosion exponentially through the cross product of their individual state spaces. In the absence of workable simulation approaches to enable virtual testing, the only recourse for verification and validation (V&V) of cyber-physical autonomous cooperative systems of systems (CACSoS) is to brute-force methods which are severely limited in the range of conditions they can test.

A key root cause of limitations in current V&V approaches to CACSoS is that they are not based on a general dynamic systems modelling and simulation framework (MSF). Such a framework should be capable of expressing the interaction of decision logic, discrete events, and continuous dynamics that are the hallmarks of such systems. We therefore propose that the discrete event system specification (DEVS) formalism, as the computational basis for a general dynamic systems theory (Zeigler et al., 2000), provides a sound and practical foundation for enhancing existing V&V methods to address their limitations in addressing in CACSoS.

The value of Modelling and Simulation in defense and other applications is well-known (Shaffer, 2012). A DEVS model is a system-theoretic concept specifying inputs, states, outputs, similar to a state machine (Mittal and Martin, 2012). Critically different however, is that it includes a time-advance function that enables it to represent discrete event systems, as well as hybrids with continuous components (Nutaro, 2011) in a straightforward platform-neutral manner (Zeigler and Sarjoughian, 2012). A recent thesis (Denil, 2013) presents a multi-paradigm model-driven approach to design, verification and deployment of software intensive systems, another formulation of cyber-physical systems. It shows that DEVS provides excellent features for modelling such systems. The thesis provides a list of properties of DEVS and their mapping to properties of automotive software and systems – here viewed as instances of CACSoS:

- *Concurrency*: Multiple processors and communication links are concurrent in a CACSoS system. The semantics of DEVS coupled models supports concurrency by appropriate interleaving of the discrete-event behaviour of individual sub-models.
- *Time*: Real-time performance is a crucial property of CACSoS embedded software. End-to-end latencies are part of the requirements for these applications. The time advance function of an atomic DEVS model can be used to model latency.
- *Events*: Event-triggered and time-triggered architectures use triggers in the form of either external events or timing events to start certain pieces of functionality. DEVS implements reaction to events using the external transition functions.
- *Priorities*: Some real-time communication channels use priority-based and other mechanisms for arbitration. DEVS supports such arbitration by means of explicit specification of executable events from the set of simultaneous events.
- *Simulation of the physical parts of the system*: DEVS is a very general formalism and is able to include different other formalisms. This generality stems from the infinite possible states that DEVS allows to model and the (continuous) time elapse between the different state transitions. The hierarchical coupling techniques are used to integrate the different formalisms using DEVS as a common denominator.

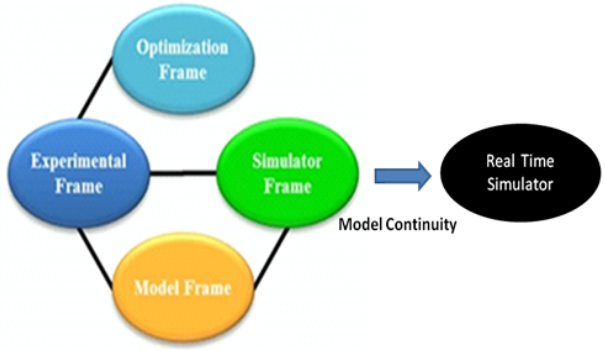
The rest of the paper is organised as follows: Section 2 provides needed background, while Section 3 discusses limitations of current V&V methods. Section 4 details DEVS support for complex system V&V followed by the integration of DEVS and non-DEVS methods in Section 5. Combining simulation and formal methods and support for this combination are discussed in Sections 6 and 7, respectively. Section 8 considers probabilistic DEVS and Markov models implementation in MS4 Me. Applications and future research are offered in Sections 9 and 10.

2 Background

The *MSF* (Zeigler, 1976) presents entities and relationships of a model and its simulation as background for the proposed work (Figure 1). The MSF separates models from simulators as entities that can be conceptually manipulated independently and then combined in a relation which defines correct simulation. The experimental frame defines a particular experimentation process, e.g., Latin hypercube

sampling for yielding model outcome measurements in accordance with specific analysis objectives. Figure 1 depicts the notion of an *optimisation frame* to supplement the MSF experimental frame, where the optimisation frame directs search among the possible models for one or more that satisfy design space criteria, including those that minimise uncertainty about how well the implemented design will work. Figure 1 also emphasises the ability enabled by model continuity to transfer a simulation model from a logical-time simulator to a real-time simulator. In particular, DEVS models for autonomous control in CACSoS can be shifted without alteration (avoiding error-prone and tedious reprogramming) to interact with real environments in cooperative configurations after being verified in virtual environments (Zeigler and Sarjoughian, 2012; Hu and Zeigler, 2008). The MSF underlies the DEVS Simulation Protocol which provides provably correct simulation execution of DEVS models thereby obviating time and state conflicts arising in simulation of multi-formalism models. There are numerous implementations of DEVS simulators (Nutaro, 2011; Mittal and Martin, 2012).

Figure 1 Model and simulation framework showing model continuity for SoS V&V (see online version for colours)



3 Limitations of V&V methods applied to CACSOS

Linear temporal logic (LTL) and computation tree logic (CTL) which are used for expressing desired behaviour and model checking have been applied to the development of vehicle routing and road monitoring in multi-UAV systems (Karaman and Frazzoli, 2008; Sirigineedi et al., 2010). Humphrey (2013) explored the use of LTL, the SPIN model checker, and the modelling language PROMELA (Gerth, 1997; Baier and Joost-Pieter, 2008; Holzmann, 2004), for high-level design and verification in UAV related applications, reporting some success while suggesting limitations and needed extensions. Table 1 shows three UAV related cases she discussed.

Table 1 Example applications of model checking to CACSoS

Model: A centralised UAV cooperator controller that coordinates the actions of multiple UAVs performing a monitoring task. *Focus of model checking:* Assuring that All sensors are eventually visited. *Sample simplifying assumptions:* Communication between UAVs and sensors can only occur when in the same location and is error free.
Model: A leader election protocol for a decentralised system of unattended ground sensors sending estimates of an intruder's position to a UAV. *Focus of model checking:* At least one leader exists at every time step. *Sample simplifying assumptions:* The sensors all use sampling epochs of the same length enabling a single time step for time advance.
Model: Verification of high level UAV mission plans for a scenario in which multiple UAVs must be used to safely escort an asset across a road network. *Focus of model checking:* The path travelled by the asset is safe, i.e., all road segments in the path have been scanned by UAV. *Sample simplifying assumptions:* UAVs and VIP were assumed to travel at the same speed.

In each case, the focus of the model is shown along with a simplifying assumption. Because they are oriented to verification, model checking tools tend to lack many functions that exist in DEVS environments and require abstractions that fit the tools' operation. This forces an abstraction of the real system that on the one hand enables the modeller to better understand the model, and on the other hand entails numerous assumptions to enable the model checker to verify the focal requirement. Despite these drastic simplifications, state space explosion prevents employing more than a handful of UAVs and sensors.

Zervoudakis et al (2013) write that

> "Research in model checking has focused on enhancing its efficiency and scalability thereby enabling model builders to verify larger, more elaborate models. Popular model checkers tend to support low-level modelling languages that require intricate models to represent even the simplest systems. For example, PROMELA, the language of the model checker SPIN, is essentially a dialect of the low-level programming language C. Another example is the modelling language used by the probabilistic model checker PRISM, whose lack of control structures forces model builders to pollute model components with counter variables that explicitly encode the components' state transitions."

These authors show that mapping of domain knowledge, assuming it exists in the right form, can be used to reduce the manual and error-prone encoding of state transitions at relatively low levels of abstraction. Here we propose to develop such mappings using the domain knowledge contained within simulation models and their ontological representations within the DEVS-based MSF.

4 DEVS Support for CACSoS

Cyber-physical systems are real-time hybrid systems, i.e., include both discrete and continuous dynamics, which, as earlier indicated, are well represented within the DEVS-based MSF. Typically such a system is described by a state consisting of both discrete control phases and continuous variables (Nutaro, 2008; Wainer, 2015), Saadawi et al. (2012) developed a methodology that combines DEVS and timed automata (TA) (Bengtsson and Yi, 2004; Henzinger, 1997; Alur, 1995; Courcoubetis et al., 1995) to allow the designer to model, simulate, verify, and deploy real-time hybrid systems. This is achieved by guaranteeing the correctness of the model with a methodology that verifies DEVS models with TA model-checking techniques and tools. Under model continuity (Figure 1) the verified DEVS models are then made executable on the target platform, thus eliminating the risk of introducing errors in the final system implementation. TA provides a solid theory and algorithms for model checking, and many existing tools implement these algorithms (e.g., UPPAAL). The combined DEVS/TA methodology deals with RTA-DEVS – a restriction of DEVS to rational (a subset of real) Time Advance values. For this subclass the methodology provides automated mappings to TA's abstract formal system specification that is verifiable by *decidable* model checking. In this methodology, if UPPAAL (or other model checker) faces a problem of state explosion, and no answers can be obtained in reasonable time, the user can use model checking on an abstraction of the system while employing DEVS-based simulations to empirically check out properties not included in the formal analysis. In particular, the methodology applies to hybrid systems whose continuous components are expressed using differential equations solved using quantised state system (QSS) integration. Concurrently, a literature is developing on the use of QSS, a class of DEVS models, to efficiently and accurately model such systems, for example using multicore processors.

While multi-agent-based simulation (ABS) is well established using DEVS (e.g., Perez et al., 2010), time-step scheduling is still used in classic ABS models (e.g., Repast, 2009). However, Zhang et al. (2014) developed a DEVS simulation model which is significantly more efficient than the Repast ABS model (350 times faster for 10,000 agents) while keeping high model spatial fidelity and the same agent cognitive capability, collision avoidance, and low agent-to-agent communication cost.

Work on non-DEVS model checking for hybrid systems includes that based on timed automata (Henzinger, 1997; Alur, 1995; Courcoubetis et al., 1995), abstraction and simplification of systems (Chauhan et al., 2002; Clarke et al., 2003; Long, 1993), and statistical model verification (Younes et al., 2006).

5 Integrating DEVS and non-DEVS verification methods

As discussed earlier, several DEVS methodologies have been developed which incorporate non-DEVS verification methods (Zeigler and Sarjoughian, 2012; Hu and Zeigler, 2008). These methodologies attempt to employ DEVS to enable loosening the simplifying assumptions typically made by non-simulation models. For example, the Sample Simplifying Assumptions in Table 1 include:

a perfect communication among component systems occur when they are in exactly specified locations

b time advances for all components use the same fixed time step

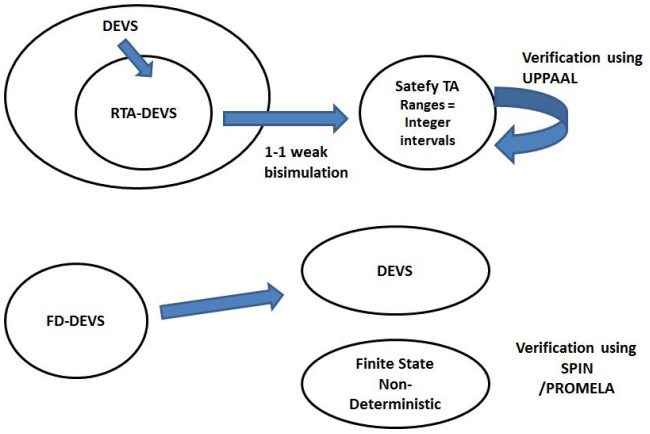
c speeds of vehicles are constant or change instantaneously.

However, so far, such methodologies have not provided a general approach to combining simulation and verification using available system theoretic concepts to relax such assumptions.

Our objective here is to develop and employ system morphisms and model transformations to integrate the various types of models to be included in a general DEVS-based framework for verification in model engineering. Model transformations are a key means of converting between different model types and must preserve desired aspects of structure and behaviour to qualify as system morphisms. Our approach is to define system morphisms for such transformations and to prove these morphisms are mathematically correct using existing theory of system morphisms (Zeigler, 1976). This will enable us eventually to *automate verification of properties for complex simulation models* as opposed to simplified models developed specifically for model checking. Then we will explore algorithmic approaches to automate the construction of these types of systems mappings. Such automation is necessary to create a practical tool for engineering systems of systems.

Figure 2 compares the methodology of Saadawi et al. (2012) which provides automated transformations from RTA-DEVS to TA enabling tractable model-checking using UPPAAL (Behrman, 2004) with the approach we discuss here. The RTA-DEVS approach seeks to verify DEVS models by transforming them into a subset of TA that can be verified using UPPAAL. To do this, it must appropriately limit the class of DEVS models to a subclass that can be mapped to the input class of TA for UPPAAL via 1-1 weak bi-simulation with the safety TA subclass of TA. Weak bi-simulation will be shown to be a system morphism. The mappings are not automated.

Figure 2 Comparing DEVS-based verification approaches (see online version for colours)



In contrast, as shown in Figure 2, our approach is to start with the FD-DEVS (Zeigler and Sarjoughian, 2012) subclass of DEVS models having finite sets of states, inputs, and outputs and whose construction is supported by MS4 Me. Then two mappings are defined:

1 map FD-DEVS models into non-deterministic automata that are subject to model checking using SPIN/PROMELA

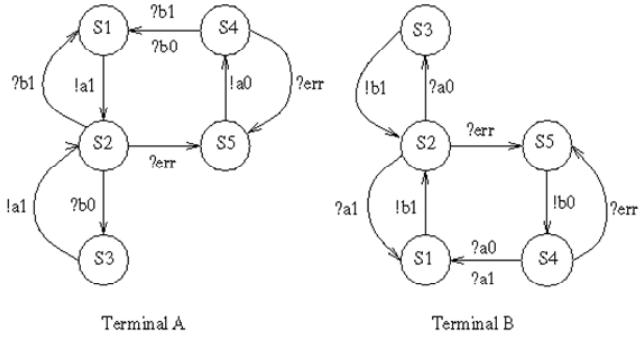
2 elaborate FD-DEVS models into full-fledged DEVS models that can be simulated to obtain behaviour of interest and possibly to discover unexpected or emergent behaviours via simulation.

Aznam et al. (2014), at about the same time as the original conference paper that gave rise to this version (Zeigler and Nutaro, 2014), discussed the complementarity between simulation and formal verification using transformation of PROMELA models into FD-DEVS models. To illustrate the general idea, we use a different example to construct an illustrative, mathematically verifiable transformation from DEVS models to model checking models and vice versa. We then employ this example in our discussion of the extension to probabilistic DEVS.

5.1 Example of system morphism between PROMELA and DEVS

For an example of this approach, consider the alternating bit protocol introduced by Bartlett et al. (1969) for implementing full-duplex communications over half-duplex communication lines. This protocol is illustrated in Figure 3. It has been used to illustrate fundamental elements and analysis capabilities of the PROMELA language by proving that the protocol operates correctly (see the SPIN manual; Gerth, 1997); that is, that "Every message fetched by A is received error-free at least once and accepted at most once by B". It is apparent from the figure that this PROMELA model is a finite state automaton, and all finite state automata are instances of DEVS models that have a fixed time advance (see Zeigler et al., 2000). Hence, a DEVS model of this protocol can be built in a simple way.

Figure 3 Model for automatic verification of ABP alternating bit protocol



We do this by adding two pieces of information to the PROMELA description: the time T to transmit a bit and the probability p of an error. One important use of this DEVS model is to answer questions about performance of the protocol. Conversely, we may map any instance of this DEVS model with parameters p and T onto a PROMELA model by abstracting away the specific probability distribution and stating only that a bit may arrive or not arrive.

In the original conference paper that gave rise to this version (Zeigler and Nutaro, 2014), we constructed a DEVS model manually to demonstrate this process. Since then we have extended the concept of FD-DEVS to a probabilistic version with supporting tools. Consequently we leave the discussion of this example to later in the paper. Before proceeding to the presentation of the extended DEVS capabilities, we note that these simulations provide important, systems level performance metrics that cannot be obtained via a query of the PROMELA model. At the same time, we may be certain that the simulation model preserves the formal properties that we have proven about the protocol by use of the PROMELA model. The capability to construct performance studies, like this bit rate study, using a simulation model derived directly from the formal verification model illustrates the power of the proposed approach for engineering complex systems.

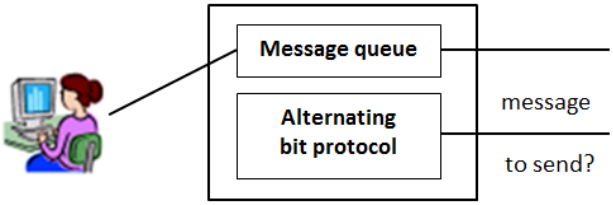
5.2 Extending the range of verification models with simulation models

Transformations between simulation models and verification models can also facilitate the reuse of model components throughout the lifecycle of an M&S system, and this reuse can have the important effect of revealing implicit assumptions in proofs constructed for the verification model. For instance, it is natural to use the alternating bit protocol as a media access control layer within a more comprehensive network simulation. This more comprehensive model could have a sender and receiver each with two components.

This is illustrated in Figure 4. The first component is our DEVS model of the alternating bit protocol. The second component sends and receives messages, rather than just bits, and queues messages that are pending transmission. This second component appends to each message the bit

that it receives from the MAC layer, and sends to the MAC layer the first bit in each message received from the network or an error indicator, as appropriate. Transmissions in the upper layer occur at the instant that a bit is received from the MAC layer below. Central to understanding the behaviour of this model is its queue capacity, rate of requests to send messages, and bit rate.

Figure 4 Message layer addition to ABP (see online version for colours)



5.2.1 Alternating bit protocol with infinite queue

Let us first consider a combination of these that may be reduced to a slightly more complex version of the verification model for the alternating bit protocol. To obtain this model we reduce the queue to two states: empty and occupied. The former indicates no messages waiting for transmission and the latter indicates a message waiting to be transmitted. Adding these states to the transmitter increases the size of the verification model from 25 states to 50 states, and we may prove for this larger model that every message transmitted is received once and only once. This proof is significant because it reflects the intended, but idealised, behaviour of the system. If, for instance, this assertion could not be proved then the design of the system should be reconsidered before moving to other forms of testing.

5.2.2 Alternating bit protocol with finite queue

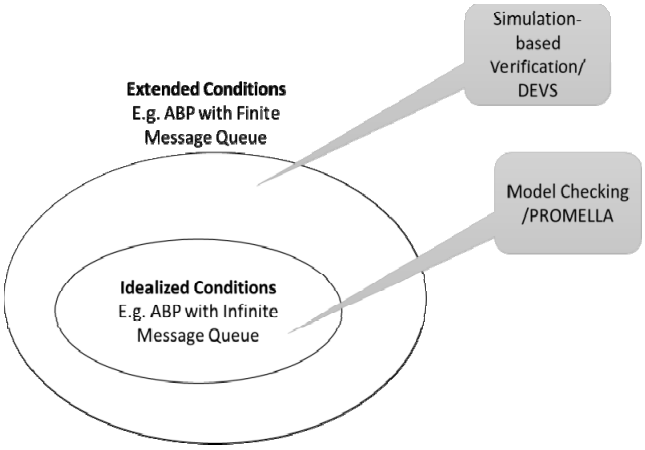
However, this proof does not indicate how the system will behave over its entire range of structural variations and realisable behaviours. In particular, we may encounter a case where the queue's capacity is finite. In this case, the bit rate and rate of requests to send messages may be such that at certain points in time the queue's capacity is exceeded as can be predicted by simple queuing theory (Jain, 1991). The consequence of this will be messages that are lost: a clear violation of the above proof! In this simple example, the cause of this violation is obvious. The verification model assumes that the queue never reaches its capacity, and so the proof implicitly assumes a restricted range of values for the queue capacity, rate of transmission requests, and bit rate.

6 Combining simulation and formal verification

These types of implicit assumptions can be difficult to identify in a large model, and simulation offers an opportunity to explore the system's parameter space and identify boundaries beyond which any particular proof fails to hold.

As illustrated in Figure 5, the combination of simulation and formal verification gives a much more powerful capability to test designs than can be achieved with either alone. In a design process that incorporates both types of analysis, verification models can be used to obtain absolute answers concerning system behaviour under idealised conditions. Failures in this verification stage clearly indicate a need to find and correct fundamental flaws in the system design.

Figure 5 Relation between verification and simulation



On the other hand, a successfully verified model can be formally extended into a simulation model for which the verification model is a homomorphic simplification. Hence, the simulation model retains the properties that were verified with the simpler model, and then can be used to explore scenarios that are necessarily outside the scope of formal verification. In some cases, other simplifications of a full-fledged simulation model can be applied (for example, as mentioned above, queueing theory can predict the probability of a finite capacity queue being exceeded). However, in general, simulation models can incorporate the complexity needed to deal with real systems using the base model concept. The traceability between these two types of models, which is obtained by the application for formal system morphisms, is central to the success of this two tiered approach to testing.

7 FD-DEVS extension to support combined simulation and verification

7.1 Finite probabilistic DEVS

Finite probabilistic DEVS (FP-DEVS) is an extension of finite deterministic DEVS (FD-DEVS) (Mittal et al., 2007) which is a foundation of DEVS Natural Language (DNL). FP-DEVS allows internal transitions out of a state to one of a finite set of possible states where the choice is made probabilistically. This relaxes the FD-DEVS rule that restricts internal transitions to a single (deterministic) transition. Starting with the FD-DEVS specification of internal transition table, the FP-DEVS extension follows:

$$InternalTransitionTable:$$
$$StateSet\ X\ ([0,1]\ UBlank) \rightarrow StateSet$$

For example, to express the probability of an internal transition from *Phase* to *Phase'* with probability *p*, we write:

$$InternalTransitionTable(Phase, p) = Phase',$$
where $0 \leq p \leq 1$

There can be several such entries, one for each destination state from the same source state; the sum of the associated probabilities is bounded by unity. If a transition back to the source is not specified, the difference between unity and the sum is attributed to the probability of the self-transition.

To express a deterministic internal transition as in the original FD-DEVS, the probability argument is left blank, as in:

$$InternalTransitionTable(PHASE, Blank)$$
$$= InternalTransitionTable(PHASE) = PHASE'$$

The *InternalTransitionTable* determines the phase of the next state in DEVS by:

1 gathering all the entries for the outgoing transitions from a source phase

2 creating a cumulative distribution function from the associated probability values

3 selecting a phase based on the distribution using a random number generator

4 setting the new phase to the one selected.

The time advance of the next state with this phase is set to the unique value associated with the internal transition.

Figure 6 Example of FP-DEVS

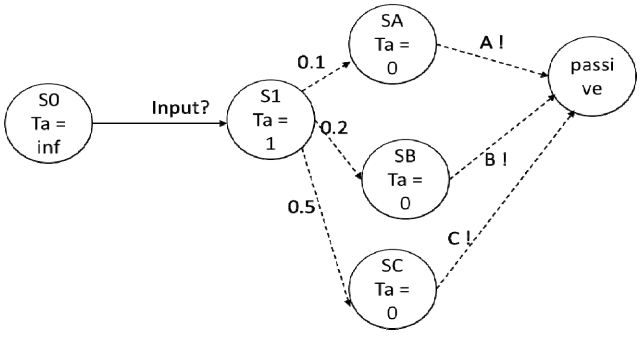


Figure 6 illustrates an example of FP-DEVS which has one input (?) and three outputs (!). S1 state can be changed to three states according to probability values at the internal transition stage. The time advances for these transitions are all the same as indicated by the Ta value for S1. Each of the target states can have any of the original specifications. So if you want to generate different outputs with given probabilities you can follow the approach of Figure 6 for outputs A, B, or C. Note that a probabilistic response to an input can also be expressed as an external transition to probabilistic source state as shown in the Figure 6.

7.2 MS4 Me software

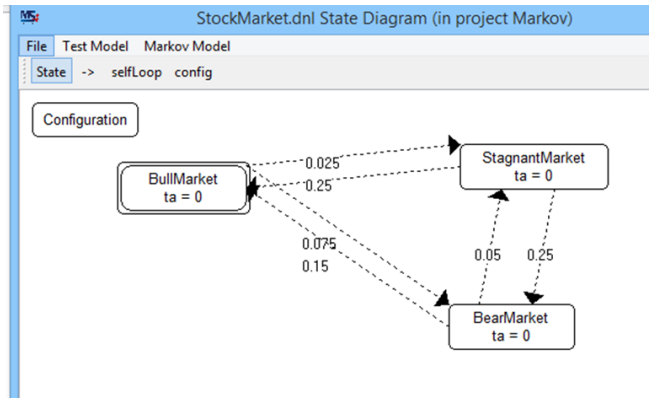
MS4 Me software (MS4 Me, 2013) implements DEVS modelling and simulation with a Java computer language in an Eclipse environment. DEVS modelling and simulation consists of atomic model (behavioural description), coupled model (model structure description), and simulation. In the MS4 Me software, the atomic model can be constructed from a state diagram designer which is a graphic user interface to help users visualise behaviours of the atomic model with symbols. Also, it can be created using a restricted DNL whose keywords are highlighted in a DNL editor in the MS4 Me software. A graphical design of an atomic model is eventually converted to a DNL file automatically transformed to a Java file for simulation. The DNL file is made of atomic model component, data component, and Java component. The atomic model component contains states, transitions, and ports (input and output). The data component displays variables and message types. The Java component covers Java features such as adding Java libraries, defining additional functions (methods), and handling (calculating) messages for which the DNL file has tag blocks enclosing Java codes for data calculation at internal transition, external transition, and output functions. The coupled model is an instance of system entity structure (SES) which describes a system with entities' relationships and coupling information in restricted natural languages. Entities represent atomic models or coupled models in the SES, and entities' relationships consist of aspect, specialisation, and multi-aspects (Zeigler, 1984). The coupling information captures message flows between entities. Theoretically, an SES with all relationships can generate an infinite number of coupled models through a pruning process which defines selections for specialisation relationships and the number of instances for multi-aspect relationships. MS4 Me provides a sequence diagram to graphically generate a SES from which associated entities are automatically converted to DNL files. The simulation is executed with a coupled Java model generated through a pruning process with a SES document and atomic Java models from DNL files. DNL and SES documents describe atomic models and coupled models. Although a DNL document contains Java specific contexts to generate a Java model, it can be utilised in other application such as web application interacting with users according to model's information.

8 Markov models: implementing morphisms in FP-DEVS

Finite state Markov chain model classes (Kemeny and Snell, 1960; Feller, 1966), with both discrete and continuous time bases, have been implemented in MS4 Me using the above described FP-DEVS capabilities. A useful example of a morphism such as discussed in Section 5.1 was developed

which maps from a *continuous time Markov* (CTM) *model* to a *discrete time Markov* (DTM) chain approximation. A second class of morphisms is illustrated by the mapping of compositions of CTMs to their components.

Figure 7 Stock market CTM model example (see online version for colours)



Source: From Wikipedia

An example of a Markov model supported by MS4 Me is the Stock Market model taken from Wikipedia (2014) as shown in Figure 7. The transition system underlying this example is interpreted as either a CTM or DTM by taking the approach of Soares and Castro (2012). Here the *continuous time* interpretation employs the probabilities associated with internal transitions as specifications of their time advances. Indeed, let the transitions out of a state (*phase*, σ) be

$$\{\tau' \mid \tau' = (phase, p', phase')\}$$

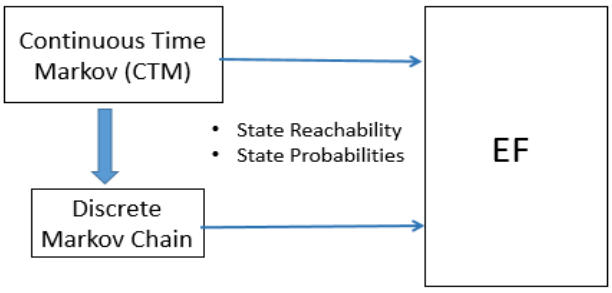
where p' is the probability of going to p*hase*' – also called the rate of transition to phase'. Then the time advance for the state is

$$\sigma_{next} = min\{\sigma' \mid \sigma' = -(1/p')ln(r)\}$$

where r is a random number between 0 and 1. The phase selected is the one whose sigma is the minimum (or one of this minimal set, if more than one exists). This implements the Markov assumption that transitions occur independently as events with Poisson distributions with exponential distribution parameters $\lambda' = 1/p'$. The model remains in *phase* for the computed time advance and transitions immediately to the selected phase. Thus the smaller is a probability of transition, the longer is the associated time to its next occurrence (on average) and the less likely it is to be selected as the transition to take (the self-transition is omitted as a contender to selection).

The *discrete time* version is parameterised by a time step (or cycle length in the common Markov terminology). As with the basic FP-DEVS convention, transitions occur with a time advance equal to the time step and with specified probabilities, in this case, determined by the given rates and the time step. For small enough time steps, the employed probabilities are given by the product of the corresponding CTM values and the time step, For larger time steps, a better approximation is given by employing probabilities equal to $1 - \exp(-h * p)$ where h is the time step and p the corresponding CTM probability (this is the probability that the first event in the Poisson process happens in the time step interval). The approximation loses accuracy with longer time steps as more than one intervening event becomes likely.

Figure 8 Experimental frame for continuous to discrete morphism (see online version for colours)



An experimental frame for the approximation of CTM by DTM models, illustrated in Figure 8, concerns preservation of state accessibility and state probabilities. A perfect approximation is one in which the same set of states is accessible from the initial state in corresponding models. Likewise, the relative times that the process dwells in the states are the same for both models. See Soares and Castro (2012) for a detailed comparison of simulation and analytic solution method results. A Markov Chain is obtained by ignoring timing and employing only the transition probabilities of the CTM. In this representation, the state is a vector of probabilities and the state trajectory becomes a sequence of state vectors on an integer time base. MS4 Me provides a class *MarkovMat* and a transformation from the FP-DEVS CTM to its MarkovMat corresponding matrix representation. A MarkovMat instance is in fact a discrete time FD-DEVS with a state vector as a state variable and an internal transition function which multiplies the vector by the matrix to update it at every time step. A test for reaching an equilibrium (stationary) state is included in the internal transition. Figure 9 depicts the convergence of the stock model Markov chain within 30 iterations. The accessible states from an initial state are computed as those with non-zero probability values in such an equilibrium state.

The mapping from a CTM to its associated MarkovMat is a structure morphism that preserves state accessibility and occupation probabilities but not residence times since no timing information is preserved (except sequencing). This is illustrated in Table 2 which shows the agreement of the equilibrium probabilities of the Markov chain with the relative occupation times observed in a run of the corresponding CTM. Due to the stochastic behaviour of the CTM, the simulation takes much longer than the matrix iteration with convergence determined by confidence interval criteria.

Figure 9 Trajectories of state vector probabilities for stock market model (see online version for colours)

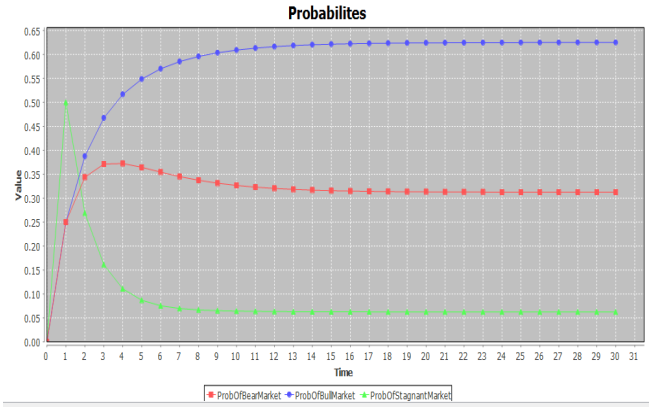


Table 2 Comparison of CTM and MarkovMat statistics

State	*CTM time in state T = 22,300*	*CTM probabilities (relative time in state)*	*MarkovMat equilibrium probabilities*
Bull	13,774	.63	.62
Stagnant	1,384	.06	.06
Bear	7,135	.31	.32
State	*CTM occupation counts (frequencies)*	*CTM avg. residence time in state*	*Sum of outgoing transition probabilities*
Bull	1,400	9.8	.1
Stagnant	686	2.0	.5
Bear	1,407	5.0	.2

Note that the occupation time in state over an interval equals the average residence time in it × the occupation count (number of times it has been entered). Sampled CTM trajectories produce such counts and therefore also average residence times as in Table 2. For consistency, such times should equal those computed as inverses of the total outgoing transition probabilities. Also Table 2 shows how the MarkovMat class plays the same role as the model checking tools described earlier in that it deals only with a limited set of properties (e.g., state accessibility) but makes them easier to compute than the more expressive super-class.

8.1 Representing lumpable Markov chains

Lumpability is a well-known condition applicable to Markov chains (Kemeny and Snell, 1960; Buchholz, 1994) that can be expressed as a structure morphism for pairs of models in the MarkovMat class. Lumpability is a kind of *aggregation uniformity condition* (Zeigler et al. 2000) that preserves state transition behaviour modulo a partition of the states. The class *BaseLumpedMarkovMat* takes a base and lumped model pair where the lumped model is constructed from a MatrixMat serving as a base model using a given state partition. The lumped probabilities are computed by averaging the elements in the inverse images of the pairs of lumped states. For example, in Figure 10, $p_{0,1} = (1/blocksize_0)\sum p_{i,j}$ where the sum is over the pairs *i, j* for which *i* maps to 0 and *j* maps to *1* and $blocksize_0$ is the cardinality of the state block containing 0.

Figure 10 Lumpability

$$P = \begin{pmatrix} \frac{1}{2} & \frac{3}{8} & \frac{1}{16} & \frac{1}{16} \\ \frac{7}{16} & \frac{7}{16} & 0 & \frac{1}{8} \\ \frac{1}{16} & 0 & \frac{1}{2} & \frac{7}{16} \\ 0 & \frac{1}{16} & \frac{3}{8} & \frac{9}{16} \end{pmatrix}$$

$$P_t = \begin{pmatrix} \frac{7}{8} & \frac{1}{8} \\ \frac{1}{16} & \frac{15}{16} \end{pmatrix}$$

Source: Example taken from Wikipedia

The lumpability condition requires that the elements in each row in the inverse images sum to the same value. The definition of the morphism just given applies to any matrix and gives the same results as lumpability when the condition holds. In fact under this condition, the mapping becomes a homomorphism from a discrete time system specification (base DTSS) to a lumped DTSS and the extension allows it to be studied as an approximate homomorphism for its error propagation behaviour (Buchholz, 1994; Zeigler et al., 2000).

8.2 Compositions of FP-DEVS and Markov models

Since FP-DEVS models are atomic DEVS models, they can be coupled to create hierarchical coupled DEVS models. MS4 Me's support of this functionality means that Markov models can be treated as ordinary DEVS models with inputs and outputs in addition to their state specifications. This resulting model can be coupled with other DEVS models, including those derived from other classes (Zeigler et al., 2000). This allows both the simulation and analytic capabilities to co-exist within the same M&S environment.

8.3 Cross-products of Markov models

Let us restrict our focus to compositions restricted to Markov models for a moment. This raises the question of closure under coupling for such model classes. For example, does coupling a pair of CTM models result in a model expressible as a CTM? Underlying this question is the zero memory property of Markov models. There are two questions:

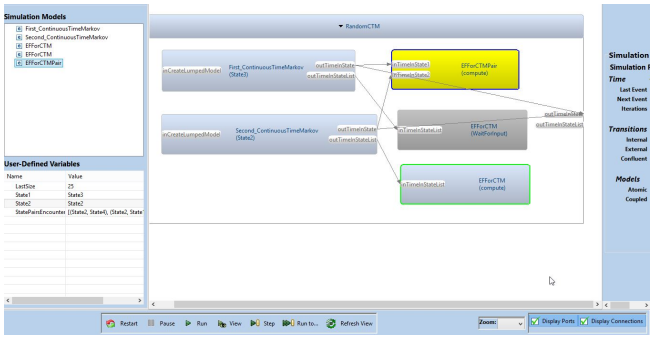
1 Is the minimal state space of the resultant of such a pair constituted by the cross product of their state sets?

2 Can the state transitions be expressed using the CTM probabilities mechanism?

Further restricting the consideration to the case where there is no cross coupling between components, we get a parallel composition or cross-product of CTMs.

Figure 11 shows a coupled model in MS4 Me composed of two randomly generated CTMs each coupled to its own experimental frame and to a joint frame, The individual frames are the same as the one in Figure 8 and track the states reported by their respective components. The joint frame however is more complex. It tracks pairs of states, where a pair gets recorded as occupied during the intervals where the individual member states overlap (since state transitions occur at random times, pairs of states do not necessarily get established in a single event).

Figure 11 Composition of CTMs with experimental frames for state reachability and residence times (see online version for colours)



The tools discussed above allow one to check whether the coupled CTM model's relative occupation times in states can be predicted by a Markov matrix model. Figure 12 illustrates how the experimental frames and models are constructed to do this. The matrices underlying the CTM models are combined in a cross product representing the joint independent combination of the Markov processes. The states of the crossproduct are the pairs i, j where i is a state of M1 and j is a state of M2. The i, j, k, l^{th} entry of the crossproduct transition matrix is set to $p1(i, k) * p2(j, l)$, the probability of independent transitions from i to k in M1 and j to l in M2 (Dayar, 2013). With this definition it is easy to see that there is a lumping morphism from the joint model to each of the component models determined by respective projection maps [the block mapping to the i^{th} state of M1 is the pair of states i, j where j ranges over the states of M2. The transition probabilities of each of these states to the block mapping to the k^{th} state sum to $p1(i, k)$]. This lumping can be viewed as a generalisation of the well-known homomorphism of parallel compositions of deterministic state systems (Zeigler, 1976). However, as indicated, while the crossproduct specifies a CTM it is not necessarily representative of the resultant of the coupled CTM model. The EF_{Pair} of Figure 12 enables us to check the predictions of the crossproduct with the actual stochastic behaviour of the CTM.

The experimental frames and models to which they are applicable as well as the derivability relationship among them are shown in Table 3. The derivability relation is explained as follows: EF_{MM} is directly applicable to MarkovMat models (in both atomic or product form) and can predict probabilities of states in equilibrium (as well as measures derivable from such probabilities such as first passage times (Kemeny and Snell, 1960). Now a CTM can serve as the base model to such a lumped model (as in Table 2) in which case, EF_{MM} is derivable from EF_{CTM} but not conversely. EF_{Pair} is needed to observe a coupled model of CTMs and can yield state occupation times. Both EF_{MM} and EF_{CTM} are derivable from EF_{Pair}.

Figure 12 Experimental frames for consistency checking of coupled CTM models (see online version for colours)

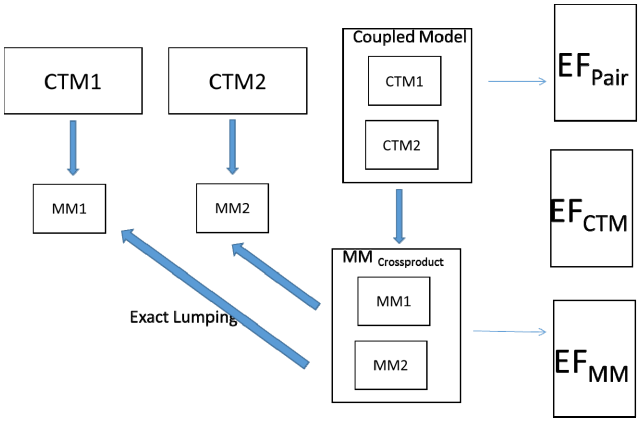


Table 3 Experimental frames and applicable models

Exp. frame	*Directly applicable to model*	*Can provide information on:*	*Derivable from:*
EF_{MM}	MarkovMat	Equilibrium state occupation times for corresponding CTM	EF_{CTM}
EF_{MM}	Crossproduct of matrices	Equilibrium state occupation times for coupled CTM	EF_{Pair}
EF_{CTM}	Continuous time Markov	State occupation times for CTM	EF_{Pair}
EF_{Pair}	Coupled CTM model	State occupation times for Coupled CTM	

8.4 *Downward and upward preservation of properties*

Zeigler and Nutaro (2015) discuss the question of upward preservation of properties from a lumped model to a base model of which it is a homomorphic image. The question is important in the context of combined simulation and model checking where it concerns whether established properties of the simplified model carry over to its complex simulation counterpart. The crossproduct of the previous section provides an example where this upward preservation demonstrably fails to hold as we now demonstrate. The property we focus on here is reachability from the initial state. It is easy to show that a state is reachable from another

in the crossproduct if, and only if, its projective images are in the same reachability relation in both components. Now, consider a case in which one of the components has unreachable states, while the other has all states reachable from the initial state. Then this case provides a counterexample to the upward preservation of connectedness for the lumping morphism: the second component has the reachability property but it is not shared by the pre-image since the other component does not. However, downward preservation of connectedness holds (it is assured by the only-if direction of the basic condition).

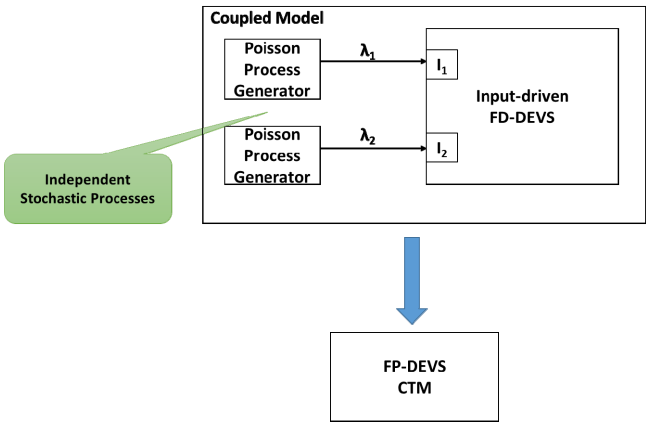
8.5 *Transformations between non-deterministic models and FP-DEVS*

Non-deterministic models such as the alternating bit protocol in Figure 3 can be directly represented in FP-DEVS. Here we note that the PROMELA model is actually deterministic except for the inputs which arrive without further specification in time. DEVS treats such unspecified arrivals as external events to a fully deterministic model.

8.5.1 *Mapping an FD-DEVS to an equivalent FP-DEVS CTM*

This leads to a general approach to mapping from an FD-DEVS atomic model to an FP-DEVS CTM model. We define an input-driven FD-DEVS as one for which every state is passive, i.e., it is time advance is infinite. Such a model waits indefinitely for input to make it transition to a next state. We discuss removing this restriction in a moment. As illustrated in Figure 13, the mapping is actually from a DEVS coupled model to the constructed FP-DEVS. The coupled model couples an independent Poisson process generator to at least one input port of the FP-DEVS to represent the arrival process of external events on that port. The added processes are independent with their own rate parameters.

Figure 13 Mapping an Input-driven FD-DEVS to an equivalent FP-DEVS CTM (see online version for colours)



The equivalence is justified as follows: having just entered a state, the FD-DEVS model waits there for the first external event to arrive (there are no internal events). This is mirrored by FP-DEVS which, in the same state, executes the time of next event method (Section 8) with rate parameters given by the ones specified for the coupled model input processes. In both cases, the transition taken is due to the earliest input to arrive. To obtain the general case in which states have finite time advances, the time of next event method can be amended to take account of the scheduled internal event by realising that that the actual event to occur is the one that occurs earliest. Thus if none of the sampled external event times is smaller than the scheduled time advance, the internal transition is executed, otherwise the earliest external transition is performed. The resulting FP-DEVS model is a modified CTM with a mixture of externally and internally derived transitions. Although the transformation has not been automated, it quite straightforward and could be done automatically, requiring only the rate parameters of the input processes from the user. In the next section, we return to the alternating bit protocol example to illustrate this construction.

9 Applications

9.1 *Performance modelling*

The tools enabled by FP-DEVS open up numerous application areas, especially where the use of stochastic modelling is novel or has been problematic in the past. The area of performance modelling is a high user of stochastic modelling (see e.g., Obaidat and Boudriga, 2010), however the combined use of verification and simulation is still under development. As a case in point, let's reconsider the alternating bit protocol in Figure 3 where we introduced parameters for the time T to transmit a bit and the probability p of an error to the PROMELA description. The FD-DEVS equivalent of this PROMELA model has three input ports – one for each binary digit and one for the error notification. Following the construction of Figure 13, we posit Poisson input generation processes where the parameter T specifies the mean parameter of the bit arrival processes, while the parameter p specifies the probability of the error arrival process. The result is a CTM expressible in FP-DEVS.

Table 4 shows the results of a simulation study, using the constructed FP-DEVS, to discover how the bit rate of the protocol varies as a function of T and p. This table was constructed with the DEVS simulation model by sweeping over a range of values for T and p, and for each combination recording the bits per second that could be exchanged by the system assuming 100% utilisation of the communication channel. The experimental frame for each combination is a transducer (Zeigler and Sarjoughian, 2012) which counts the number of bits produced at the output of the model in the observation time interval. Interestingly, because it relies on bit identities, this *throughput frame is not derivable from*

any of the frames associated with Markov models discussed above. Executing the model produces results as in Table 4 which confirms that the throughput is directly proportional to the arrival rate ($1/T$) and to the probability of correct transmission ($1 - p$). The MarkovMat matrix model which is directly derivable from the CTM model offers some verification help in that it predicts reachability of states. For example, it confirms that states S4 and S5 are not reachable from the initial state when the error probability is zero. Although obvious by inspection in this simple example, this approach does offer a fast reachability check for more complex models.

These simulations provide important systems level performance metrics that cannot be obtained via a query of the PROMELA model. At the same time, if upwards preservation is established, the simulation model may also have the formal properties that can be proven about the protocol by use of the PROMELA model. *The capability to construct performance studies, like this throughput study, using a simulation model derived directly from the formal verification model illustrates the power of the proposed approach for engineering complex systems*.

Table 4 Performance study of the alternating bit protocol

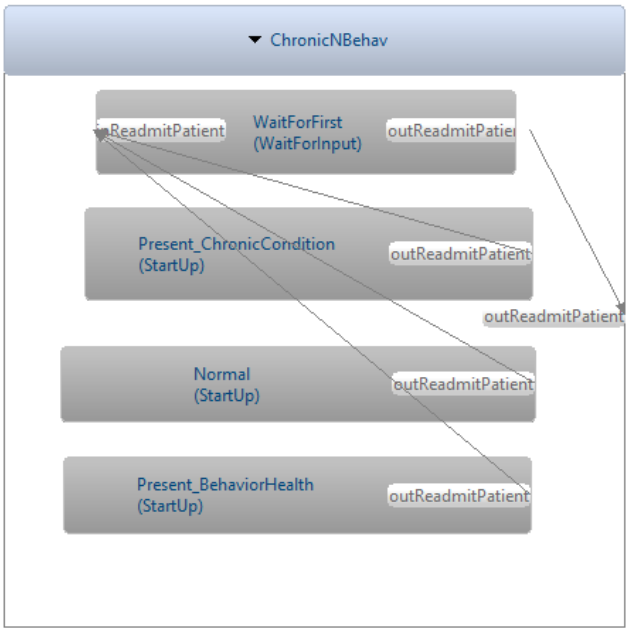
T (seconds)	*P*	*Bits per second*	*T (seconds)*	*P*	*Bits per second*
1E–9	0	5.0E8	1E–6	0	5.0E5
	0.25	3.8E8		0.25	3.8E5
	0.5	2.5E8		0.5	2.5E5

9.2 Healthcare intervention quality/cost tradeoffs

Another illustrative direct application is to evaluation of cost effectiveness of healthcare interventions. There is an ongoing controversy in evaluating the tradeoff of quality of life with cost in drug therapy. This controversy relates to the use of Markov modelling versus what they call Discrete Event Simulation (DES). Karnon and Afzali (2014) review a large number of studies in the area and discuss pros and cons of the use of the first vs the second. Basically Markov models are also called cohort models which follow patients as a group over time in aggregated form. Consequently, these models cannot accommodate patient differences However, they are easier to develop and calibrate and run very fast. In contrast, DES requires more detailed data, are much slower to execute and require numerous runs to achieve statistical validity. However, they can easily accommodate individual patient stochastic disease and treatment trajectories. Rather than either-or use of Markov and DES, FP-DEVS allows both approaches to be used in an integrated manner. The Markov model classes CTM, DTM, and MarkovMat support cohort level models where aggregations of homogeneous patients are tracked. Moreover, the embedding of CTM and DTM models within FP-DEVS enables them to serve within patient models to produce the individualised stochastic and disease trajectories of DES.

Figure 14 illustrates a coupled model with CTMs representing processes that predict readmission to hospital after discharge for patients with normal, chronic disease and behavioural health issues. One or both of the chronic and behavioural CTM model components are pruned depending on an individual patient's attributes. The output, signalling the patient's readmission to hospital, is the earliest output received by the WaitForFirst component from the CTM risk of readmission models. Moreover, consistency checking and cross-validation between the two levels of representation are supported by the morphisms and bi-simulation classes (e.g., BaseLumpedMarkovMat) discussed above. This integration obviates the need to make the either-or choice assumed in some healthcare effectiveness evaluation communities.

Figure 14 FP-DEVS coupled model of components that can be pruned to represent individual attributes (see online version for colours)



9.3 Physics

Markov chain models have been studied in physics under the name 'interacting particle system', an example of which is the well-known Ising model, a stochastic cellular automaton. The generalisation of FD-DEVS to FP-DEVS follows the spirit of the generalisation from deterministic cellular spaces to stochastic versions (Ligget, 1997).

9.4 Ballistic missile defense

Ballistic missiles follow a four-phased trajectory path: boost, ascent, midcourse, and terminal. Since systems of systems to detect and intercept such missiles cannot be tested at the mission level, developing trustworthy abstract models is desired (Mittal and Rainey, 2015). Similar to disease stages in healthcare, each of the missile phases can

be represented by families of FP-DEVS models and coupled together at different levels of granularity.

10 Future research

Future research must develop robust transformation techniques together with proofs that show them to be systems morphisms of suitable types. This will ensure consistency between the results of verification of emergent foreseen behaviours using computational analytical techniques and the discovery of emergent unforeseen behaviours through dynamic simulations. Such techniques will require formulating a meta-modelling approach that will lead to a multi-step verification process that can handle the dynamical complexity of CACSoS. We recognise that the most expeditious way to develop an inclusive framework is to build on existing methods and software tools to the extent possible. The multi-step verification process will help to manage and cross-check results obtained from the various analytical and simulation methods, as well as from integrated methods that are supported by the proposed framework. This approach helps to deal with the complexities of CACSoS by enabling more robust designs and a more thorough and organised simulation and verification process. These complexities can be addressed with the help of the theory of M&S (Zeigler, 1976) to develop methods for hierarchical decomposition and simplification as essential tools within the emerging field of model engineering (Zhang et al., 2014).

References

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J. and Yovine, S. (1995) 'The algorithmic analysis of hybrid systems', *Theoretical Computer Science*, Vol. 138, No. 1, pp.3–34.

Baier, C. and Joost-Pieter, K. (2008) *Principles of Model Checking*, The MIT Press, Cambridge, MA.

Bartlett, K.A., Scantlebury, R.A. and Wilkinson, P.T. (1969) 'A note on reliable full-duplex transmission over half-duplex lines', *Comm. of the ACM*, Vol. 12, No. 5, pp.260–265.

Behrmann, G., David, A. and Larsen, K. (2004) 'A tutorial on UPPAAL', *Proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, LNCS*, Vol. 3185.

Bengtsson, J. and Yi, W. (2004) 'Timed automata: semantics, algorithms and tools', *Lectures on Concurrency and Petri Nets*, Vol. 3098.

Buchholz, P. (1994) 'Exact and ordinary lumpability in finite Markov chains', *J. Appl. Probab.*, Vol. 31, pp.59–75.

Chauhan, P., Clarke, E., Kukula, J., Sapra, S., Veith, H. and Wang, D. (2002) 'Automated abstraction refinement for model checking large state spaces using SAT based conflict analysis', in Aagaard, M.D. and O'Leary, J.W. (Eds.): *FMCAD*, *LNCS*, Vol. 2517, pp.33–51, Springer, Heidelberg.

Clarke, E., Grumberg, O., Jha, S., Lu, Y. and Veith, H. (2003) 'Counterexample-guided abstraction refinement for symbolic model checking', *Journal of the ACM (JACM)*, Vol. 50, No. 5, pp.752–794.

Dayar, T. (2013) *Analyzing Markov Chains using Kronecker Products: Theory and Applications*, Springer, Briefs in Mathematics, Berlin, Germany, DOI: 10.1007/978-1-4614-4190-86.

Denil, J. (2013) *Design, Verification and Deployment of Software Intensive Systems: A Multi-Paradigm Modelling Approach*, PhD dissertation, University of Antwerp.

Feller, W. (1966) *An Introduction to Probability Theory and its Applications*, pp.1–2, Wiley, New York, NY.

Gerth, R. (1997) *Concise PROMELA Reference* [online] http://SPINroot.com/SPIN/Man/Quick.html (accessed July 2014).

Henzinger, T.A., Ho, P.H. and Wong-Toi, H. (1997) 'HyTech: a model checker for hybrid systems', *International Journal on Software Tools for Technology Transfer (STTT)*, Vol. 1, No. 1, pp.110–12.

Holzmann, G.J. (2004) *The SPIN Model Checker: Primer and Reference Manual*, Addison Wesley Publishing Company, Boston, MA.

Hu, X. and Zeigler, B.P. (2005) 'A simulation-based virtual environment to study cooperative robotic systems', *Integrated Computer-Aided Engineering (ICAE)*, Vol. 12, No. 4, pp.353–367.

Humphrey, L.R. (2013) 'Model checking for verification in UAV cooperative control applications recent advances in research on unmanned aerial vehicles', *Lecture Notes in Control and Information Sciences*, Vol. 444, pp.69–117.

Jain, R.K. (1991) *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling Hardcover*, 685 pp., Wiley, New York, NY.

Karaman, S. and Frazzoli, E. (2008) 'Vehicle routing with linear temporal logic specifications: applications to multi-UAV mission planning', in *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*.

Karnon, J. and Afzali, H.H.A. (2014) 'When to use discrete event simulation (DES) for the economic evaluation of health technologies? A review and critique of the costs and benefits of DES', *PharmacoEconomics*, Vol. 32, pp.547–558.

Kemeny, J.G. and Snell, J.L. (1960) *Finite Markov Chains*, v. Nostrand, Princeton, NJ.

Liggett, T.M. (1997) 'Stochastic models of interacting systems', *The Annals of Probability*, Vol. 25, No. 1, pp.1–29, Institute of Mathematical Statistics.

Long, D.E. (1993) *Model Checking, Abstraction, and Compositional Verification*, PhD thesis, Carnegie Mellon University.

Mittal, S. and Martin, J.L.R. (2012) *Netcentric System of Systems Engineering with DEVS Unified Process*, 1st ed., CRC Press, Boca Raton, FL.

Mittal, S. and Rainey, L.B. (2015) 'Engineering emergence in system of systems: the ballistic, missile defense system as a case study', *Journal of Defense Modeling and Simulation*, (accepted for publication).

Mittal, S., Zeigler, B.P. and Hwang, M.H. (2007) *XFD-DEVS* [online] http://www.duniptechnologies.com/research/xfddevs/ (accessed June 2015).

MS4 Me (2013) *MS4 Me Software V 3.0 (Markov Modeling Capability)* [online] http://www.ms4systems.com.

Nutaro, J. (2008) 'On constructing optimistic simulation algorithms for the discrete event system specification', *ACM Transactions on Modeling and Computer Simulation*, Vol. 19, No. 1, pp.1–21.

Nutaro, J. (2011) *Building Software for Simulation: Theory and Algorithms with Applications in C++*, Wiley, New York, NY.

Obaidat, M.S. and Boudriga, N. (2010) *Fundamentals of Performance Evaluation of Computer and Telecommunications Systems*, John Wiley & Sons, ISBN: 0471269832, New York, NY.

Perez, E., Ntaimo, L., Bailey, C. and McCormack, P. (2010) 'Modeling and simulation of nuclear medicine patient service management in DEVS', *Simulation-Transactions of the Society for Modeling and Simulation International*, Vol. 86, Nos. 8–9, pp.481–501.

Repast (2009) *Repast Home Page* [online] http://repast.sourceforge.net (accessed July 2014).

Saadawi, H., Wainer, G. and Moallemi, M. (2012) 'Principles of models verification for real-time embedded applications', in K. Popovici and P. Mosterman (Eds.): *Real-Time Simulation Technologies: Principles*, *Methodologies*, *and Applications*, Taylor and Francis, CRC Press.

Seo, C., Zeigler, B.P., Coop, R. and Kim, D. (2013) 'DEVS modeling and simulation methodology with MS4 Me software tool', *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium (TMS-DEVS)*, April, San Diego, CA.

Shaffer, A.R. (2012) 'The value of modeling and simulation for the Department of Defense', *M&S Journal*, Fall 2012, pp.2–3.

Sirigineedi, G., Tsourdos, A., White, B. and Zbikowski, R. (2010) 'Kripke modelling and model checking of a multiple UAV system monitoring road network', in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*.

Soares, M.O. and Castro, L.L.C. (2012) 'Continuous time simulation and, discretized models for cost-effectiveness analysis', *Pharmacoeconomics*, 1 December, Vol. 30, No. 12, pp.1101–1117, doi: 10.2165/11599380-000000000-00000..

Wainer, G.A. (2015) 'The cell-DEVS formalism as a method for activity tracking in spatial modelling and simulation', *Int. J. of Simulation and Process Modelling*, Vol. 10, No. 1 pp.19–38.

Wikipedia (2014) [online] http://en.wikipedia.org/wiki/Markov_chain (accessed July 2014).

Yacoub, A., Hamri, M. and Frydman, C. (2014) 'Complementarity between simulation and formal verification, transformation of PROMELA models into FDDEVS models: application to a case study', *Simultech*.

Younes, H.L.S., Kwiatkowska, M., Norman, G. and Parker, D. (2006) 'Numerical vs. statistical probabilistic model checking', *International Journal on Software Tools for Technology Transfer (STTT)*, Vol. 8, No. 3, pp.216–228.

Zeigler, B.P. (1976) *Theory of Modeling and Simulation*, 1st ed., John Wiley & Sons, New York, NY.

Zeigler, B.P. (1984) *Multifaceted Modeling and Discrete Event Simulation*, Academic Press, London, UK.

Zeigler, B.P. (2014) 'The role of modeling and simulation in coordination of health care', *Proceedings of Simultech*, Vienna, Video [online] http://vimeo.com/105849693.

Zeigler, B.P. and Nutaro, J. (2014) 'Combining DEVS and model-checking: using systems morphisms for integrating simulation and analysis in model engineering', *EMSS: European Modelling & Simulation Symposium*, 26th ed., Part of I3M2014, Bordeaux, France, 10–12 September 2014.

Zeigler, B.P. and Nutaro, J. (2015) 'Towards a framework for more robust validation and verification of simulation models for systems of systems', *JDMS*, 26 February 2015, doi: 10.1177/1548512914568657 (to be published 2016).

Zeigler, B.P. and Sarjoughian, H.S. (2012) *Guide to Modeling and Simulation of Systems of Systems*, p.393, Springer, Berlin, Germany.

Zeigler, B.P., Praehofer, H. and Kim, T.G. (2000) *Theory of Modeling and Simulation*, 2nd ed., Academic Press, London, UK.

Zervoudakis, F., Rosenblumy, D.S., Elbaumz, S. and Finkelstein, A. (2013) 'Cascading verification: an integrated method for domain-specific model checking', *ESEC/FSE*, Saint Petersburg, Russia.

Zhang, B., Chan, W.K.V. and Ukkusuri, S.V. (2014) 'On the modelling of transportation evacuation: an agent-based discrete-event hybrid-space approach', *Journal of Simulation*, Vol. 8, No. 4, pp.259–270, doi:10.1057/jos.2014.

Zhang, L., Shen, Y.W., Zhang, X.S., Song, X., Tao, F. and Liu, Y. (2014) 'The model engineering for complex system simulation', *The 26th European Modeling & Simulation Symposium (Simulation in Industry)*, Bordeaux, France, September 10–12.